1. Binary Search Using Divide And Conquer

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int n,i,arr[50],search,first,last,middle;
cout<<"enter total of elements";
cin>>n;
cout<<"enter"<<n<<"number\n";
for(i=0;i<n;i++)
{
cin>>arr[i];
}
cout<<"enter a number to find";
cin>>search;
first=0;
last=n-1;
middle=(first+last)/2;
while(first<=last)
{
  if(arr[middle]<search)
  {
    first=middle+1;
    }
    else if(arr[middle]==search)
    {
    cout<<search<<"found at location"<<middle+1<<"\n";
    break;
    }
    else
    {
            last=middle-1;
            }
            middle=(first+last)/2;
            }
            if(first>last)
            {
            cout<<"not found"<<search<<"is not present in the list";
            }
            getch();
            }
```

OUTPUT:


enter total of elements 5
enter5number
45
65
66
39
82
enter a number to find 66
66found at location3

2. Quick Sort using Divide and Conquer

```
#include<iostream.h>
#include<conio.h>
void qsort(int arr[],int low,int high);
int partition(int arr[],int low,int high);
void swap(int *x,int *y)
{
int temp=*x;
*x=*y;
*y=temp;
}
void qsort(int arr[],int low,int high)
{
 if(low<high)
 {
 int pi=partition(arr,low,high);
 qsort(arr,low,pi-1);
 qsort(arr,pi+1,high);
 }
  }
int partition(int arr[],int low,int high)
{
int i,j,pivot;
pivot=arr[high];
i=(low-1);
for(j=low;j<=high-1;j++)
{
if(arr[j]<=pivot)
{
 i++;
 swap(&arr[i],&arr[j]);
 }
}
swap(&arr[i+1],&arr[high]);
return(i+1);
}
int main()
{
int n;
clrscr();
cout<<"enter the number:";
cin>>n;
int a[50];
int i;
for(i=0;i<n;i++)
cin>>a[i];
qsort(a,0,n-1);
i=0;
cout<<"sort array:\n";
while(i<n)
{
```

```
cout<<a[i]<<endl;
i++;
}
getch();
return 0;
}
```

OUTPUT:

3. Merge Sort using Divide and Conquer

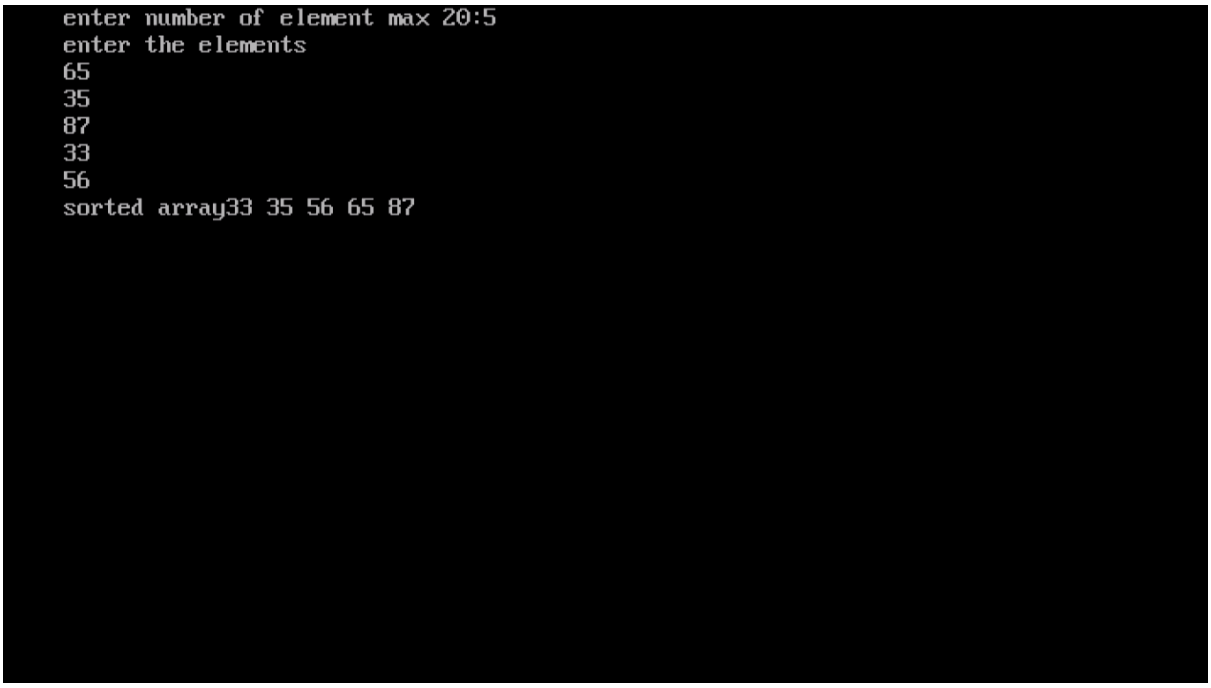
```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int start1,int end1,int start2,int end2);
int main()
{
clrscr();
int array[20],num,i;
cout<<"enter number of element max 20:";
cin>>num;
cout<<"enter the elements";
cout<<"\n";
 for(i=0;i<num;i++)
 cin>>array[i];
 mergesort(array,0,num-1);
 cout<<"sorted array";
 for(i=0;i<num;i++)
 cout<<array[i]<<" ";
 getch();
 return 0;
 }
 void mergesort(int a[],int p,int q)
 {
 int mid;
 if(p<q)
 {
 mid=(p+q)/2;

 mergesort(a,p,mid);
 mergesort(a,mid+1,q);
 merge(a,p,mid,mid+1,q);
 }
 }
 void merge(int a[],int start1,int end1,int start2,int end2)
 {
 int temp[50];
 int i,j,k=0;
 i=start1;
 j=start2;

 while(i<=end1 && j<=end2)
 {
 if(a[i]<a[j])
 temp[k++]=a[i++];
 else
 temp[k++]=a[j++];
 }
 while(i<=end1)
 temp[k++]=a[i++];
```

```
while(j<=end2)
temp[k++]=a[j++];
for(i=start1,j=0;i<=end2;i++,j++)
a[i]=temp[j];
}
```

OUTPUT:


enter number of element max 20:5
enter the elements
65
35
87
33
56
sorted array33 35 56 65 87

4. Selection Sort using Divide and Conquer

```
#include<iostream.h>
#include<conio.h>
void swap(int &x,int &y)
{
int temp=x;
x=y;
y=temp;
}
void sort(int num[],int n)
{
int i,j,min;
for(i=0;i<n-1;++i)
{
min=i;
for(j=i+1;j<n;++j)
if(num[j]<num[min])
min=j;
swap(num[min],num[i]);
}
}
int main()
{
int n;
clrscr();
cout<<"ENTER THE NUMBER:";
cin>>n;
int a[50];
int i;
for(i=0;i<n;i++)
cin>>a[i];
sort(&a[0],n);
i=0;
cout<<"SORTED ARRAY:\n";
while(i<n)
{
cout<<a[i]<<endl;
i++;
}
getch();
return 0;
}
```

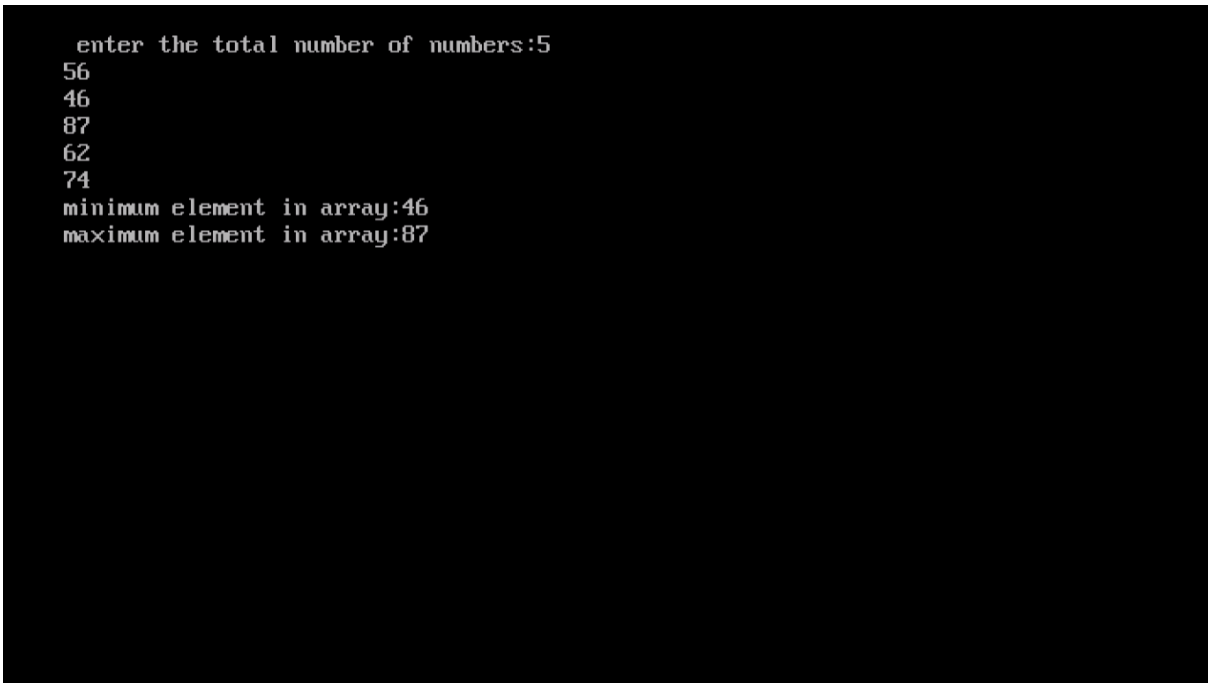
OUTPUT:

5. Maximum and Minimum using Divide using Conquer

```
#include<iostream.h>
#include<conio.h>
int max,min;
int a[100];
void maxmin (int,int);
void main()
{
int i,num;
clrscr();
cout<<"\n enter the total number of numbers:";
cin>>num;
for(i=1;i<=num;i++)
{
cin>>a[i];}
max=a[0];
min=a[0];
maxmin(1,num);
cout<<"minimum element in array:"<<min<<endl;
cout<<"maximum element in array:"<<max<<endl;
getch();
}
void maxmin (int i,int j)
{
int max1,min1,mid;
if(i==j)
{
max=min=a[i];
}
else
{
 if(i==j-1)
 {
 if(a[i]<a[j]){
 max=a[j];
 min=a[i];
 }
 else
 {
 max=a[i];
 min=a[j];
 }
 }
 else
 {
 mid=(i+j)/2;
 maxmin(i,mid);
 max1=max;
 min1=min;
 maxmin(mid+1,j);
 if(max<max1)
```

```
  max=max1;
  if(min>min1)
  min=min1;
  }
 }
}
```

OUTPUT:

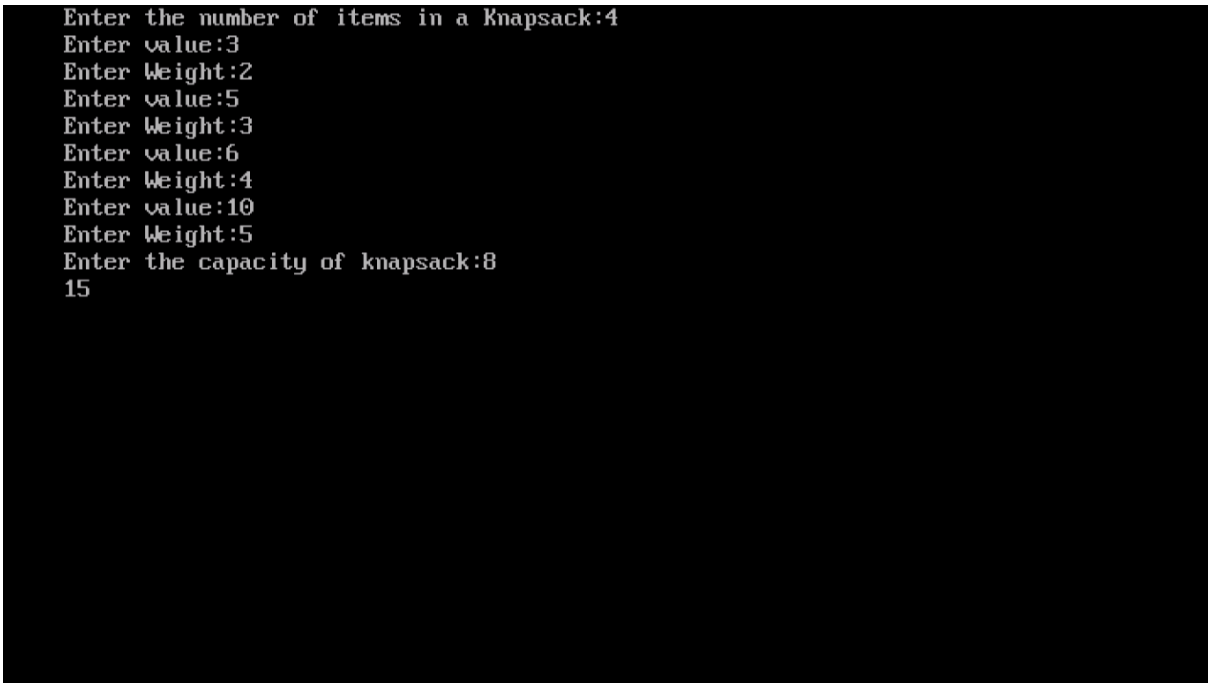

enter the total number of numbers:5
56
46
87
62
74
minimum element in array:46
maximum element in array:87

6. 0/1 Knapsack using Dynamic Programming

```
#include<iostream.h>
#include<conio.h>
int max(int a,int b)
{
return(a>b)?a:b;
}
int knapsack(int W,int wt[],int val[],int n)
{
if(n==0||W==0)
return 0;
if(wt[n-1]>W)
return knapsack(W,wt,val,n-1);
else return max(val[n-1]+knapsack(W-wt[n-1],wt,val,n-1),knapsack(W,wt,val,n-1));
}
int main()
{
int n,W;
clrscr();

cout<<"Enter the number of items in a Knapsack:";
cin>>n;
int val[20],wt[20];
for(int i=0;i<n;++i)
{
cout<<"Enter value:";
cin>>val[i];
cout<<"Enter Weight:";
cin>>wt[i];
}
cout<<"Enter the capacity of knapsack:";
cin>>W;
cout<<knapsack(W,wt,val,n);
getch();
return 0;
}
```

OUTPUT:


Enter the number of items in a Knapsack:4
Enter value:3
Enter Weight:2
Enter value:5
Enter Weight:3
Enter value:6
Enter Weight:4
Enter value:10
Enter Weight:5
Enter the capacity of knapsack:8
15

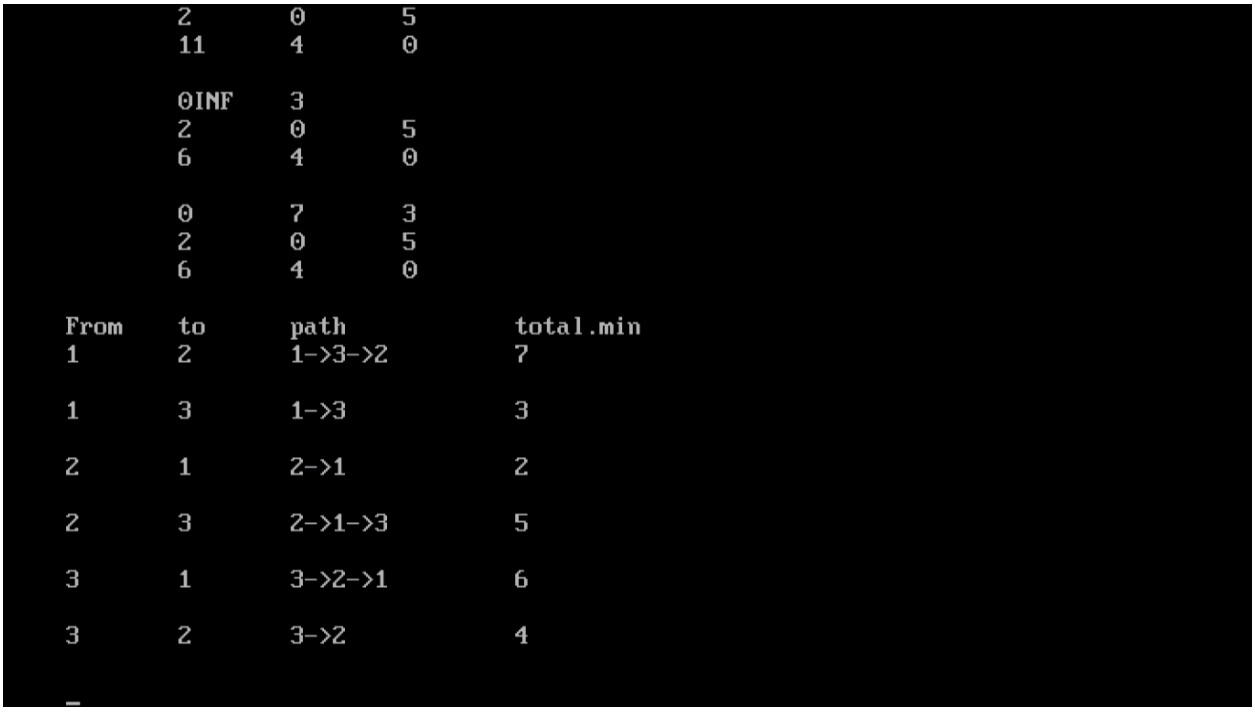
7. All pairs of shortest path algorithm

```
#include<iostream.h>
#include<conio.h>
int c[100][100],p[100][100];
int inf=1000,v,m;
void show();
void path(int,int);
int main()
{
int i,j,k,x;
clrscr();
cout<<"Enter the no of vertices:";
cin>>v;
cout<<"Enter adjacency matrix:\n";
cout<<"If the value is inf enter 1000:\n";
for(i=1;i<=v;i++)
for(j=1;j<=v;j++)
{
cin>>c[i][j];
p[i][j]=-1;
}
cout<<"\n";
for(k=1;k<=v;k++)
{
for(i=1;i<=v;i++)
{
for(j=1;j<=v;j++)
{
if(i==j)
        c[i][j]=0;
        else
        {
x=c[i][k]+c[k][j];
if(c[i][j]>x)
        {
        c[i][j]=x;
        p[i][j]=k;
        }
                }
                }
                }
        show();
        cout<<"\n";
        }
        cout<<"From\tto\tpath\t\ttotal.min";
        for(i=1;i<=v;i++)
        {
        for(j=1;j<=v;j++)
        {
if(i!=j)
```

```
{
cout<<"\n";
cout<<i;
cout<<"\t"<<j;
cout<<"\t"<<i;
path(i,j);
cout<<j<<"\t\t";
cout<<c[i][j];
cout<<"\n";
}
}
}
cout<<"\n";
getch();
return 0;
}
void show()
{
int i,j;
for(i=1;i<=v;i++)
        {
        for(j=1;j<=v;j++)
        if(c[i][j]==1000)
        {
        cout<<"INF";
        }
else
        {
        cout<<"\t"<<c[i][j];
        }
cout<<"\n";
}
}
void path(int i,int j)
{
int k;
k=p[i][j];
if(k==-1)
{
cout<<"->";
return;
}
path(i,k);
cout<<k;
path(k,j);
}
```

OUTPUT:


Enter the no of vertices:3
Enter adjacency matrix:
If the value is inf enter 1000:
0 1000 3
2 0 6
11 4 0

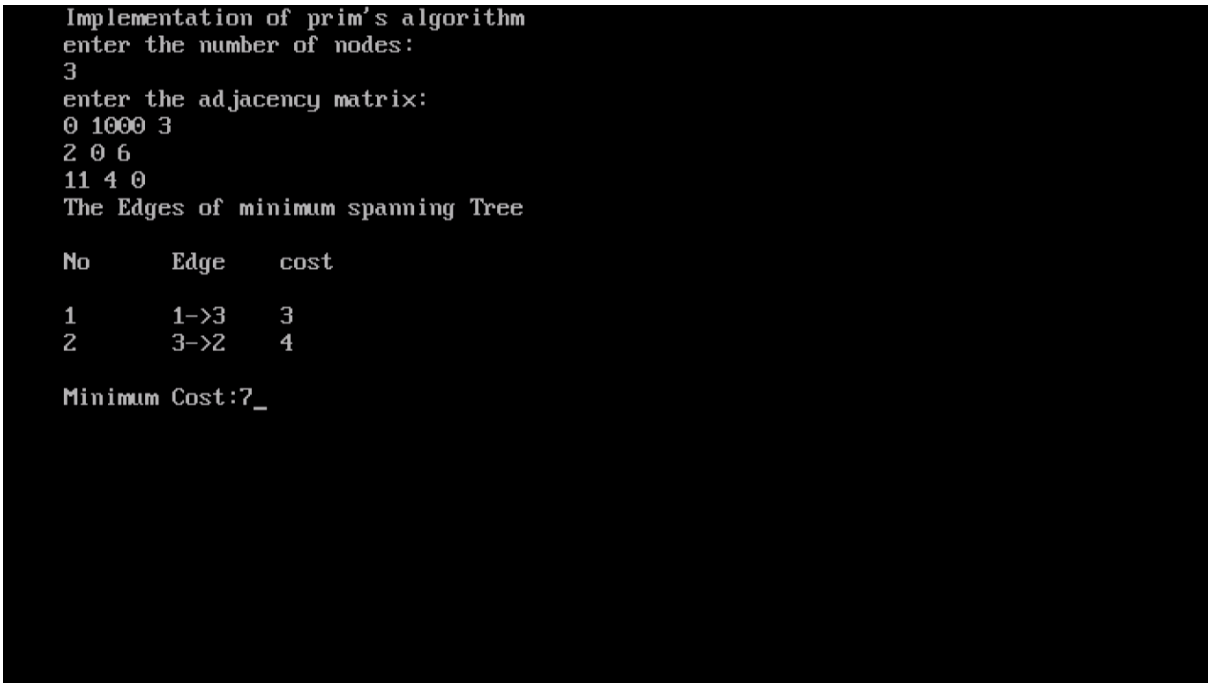

2 0 5
11 4 0
0INF 3
2 0 5
6 4 0
0 7 3
2 0 5
6 4 0
From to path total.min
1 2 1->3->2 7
1 3 1->3 3
2 1 2->1 2
2 3 2->1->3 5
3 1 3->2->1 6
3 2 3->2 4

8. Minimum Cost Spanning Tree using Prims Algorithm & Kruskal Algorithm

A) Prims Algorithm:

```
#include<iostream.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited [10]={0},min,mincost=0,cost[10][10];
void main()
{
clrscr();
cout<<"Implementation of prim's algorithm\n";
cout<<"enter the number of nodes:\n";
cin>>n;
cout<<"enter the adjacency matrix:\n";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
cin>>cost[i][j];
if(cost[i][j]==0)
cost[i][j]=999;
}
visited[1]=1;
cout<<"The Edges of minimum spanning Tree\n";
cout<<"\nNo\tEdge\tcost\n";
while(ne<n)
{
for(i=1,min=99;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]<min)
if(visited[i]!=0)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
if(visited[u]==0||visited[v]==0)
{
cout<<"\n"<<ne++<<"\t"<<a<<"->"<<b<<"\t"<<min;
mincost+=min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
cout<<"\n\nMinimum Cost:"<<mincost;
getch();
}
```

OUTPUT:

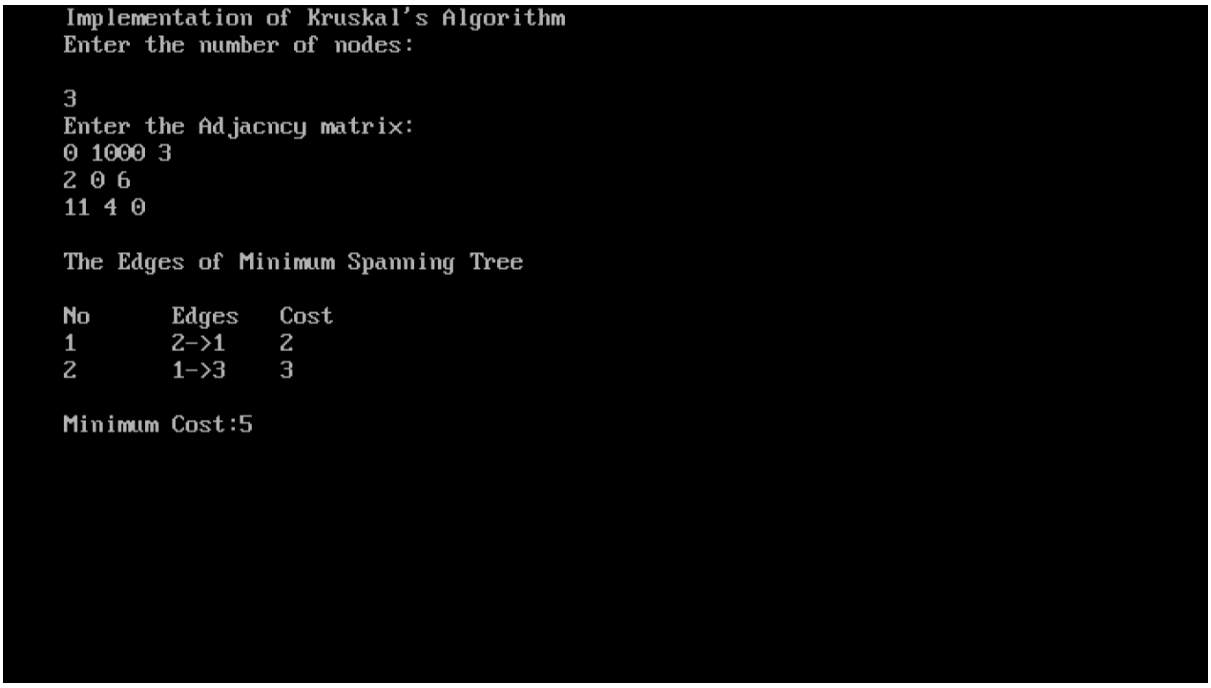

Implementation of prim's algorithm
enter the number of nodes:
3
enter the adjacency matrix:
0 1000 3
2 0 6
11 4 0
The Edges of minimum spanning Tree
No Edge cost
1 1->3 3
2 3->2 4
Minimum Cost:7_

B) Kruskal Algorithm

```
#include<iostream.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int parent[9],min,mincost=0,cost[10][10];
int find(int);
int uni(int,int);
void main()
{
clrscr();
cout<<"Implementation of Kruskal's Algorithm\n";
cout<<"Enter the number of nodes:\n";
cin>>n;
cout<<"Enter the Adjacncy matrix:\n";
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
cin>>cost[i][j];
if(cost[i][j]==0)
cost[i][j]=999;
}
}
cout<<"\n";
cout<<"The Edges of Minimum Spanning Tree\n\n";
cout<<"No\tEdges\tCost";
while(ne<n)
{
for(i=1,min=999;i<=n;i++)
{
for(j=1;j<=n;j++)
if(cost[i][j]<min)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
}
if(uni(u,v))
{
cout<<"\n"<<ne++<<"\t"<<a<<"->"<<b<<"\t"<<min;
mincost+=min;
}
cost[a][b]=cost[b][a]=999;
}
cout<<"\n\nMinimum Cost:"<<mincost;
getch();
}
int find(int i)
{
```

```
while(parent[i])
i=parent[i];
return i;
}
int uni(int i,int j)
{
if(i!=j)
{
parent[j]=i;
return 1;
}
return 0;
}
```

OUTPUT:

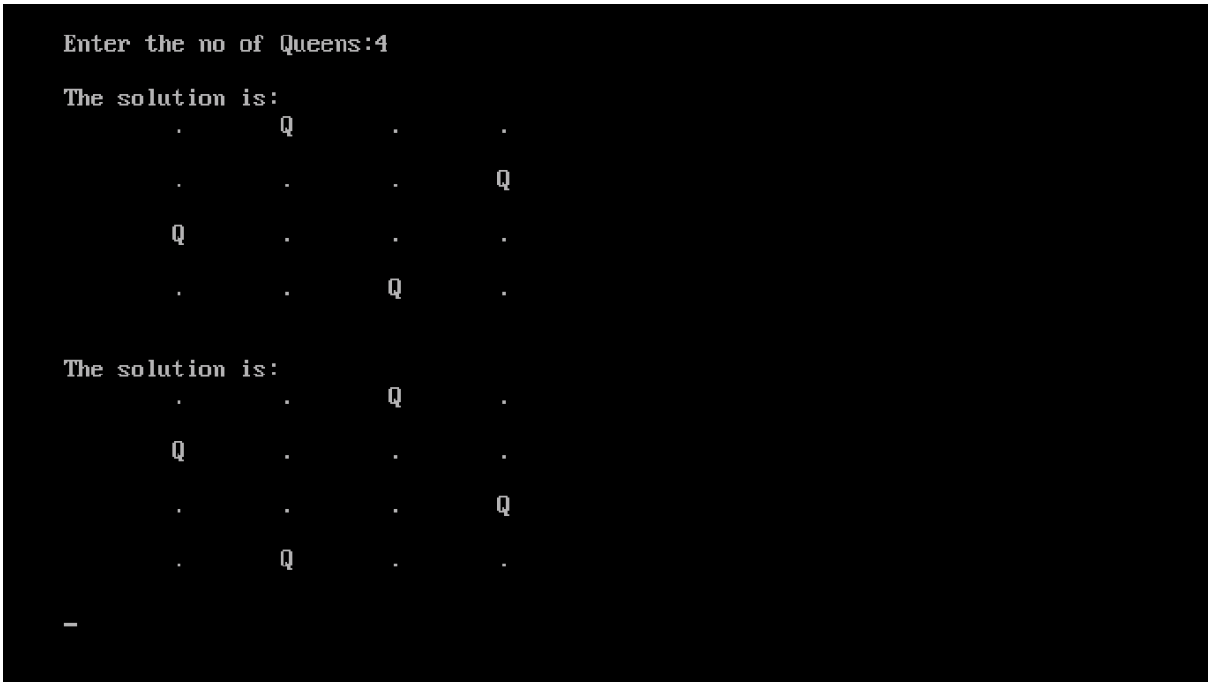

Implementation of Kruskal's Algorithm
Enter the number of nodes:
3
Enter the Adjacncy matrix:
0 1000 3
2 0 6
11 4 0
The Edges of Minimum Spanning Tree
No Edges Cost
1 2->1 2
2 1->3 3
Minimum Cost:5

9. N Queens Problem using Backtracking

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
char a[10][10];
int n;
void printmatrix()
{
int i,j;
cout<<"\n";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
cout<<"\t"<<a[i][j];
cout<<"\n\n";
}
}
int getmarkedcol(int row)
{
int i;
for(i=0;i<n;i++)
if(a[row][i]=='Q')
{
return(i);
break;
}
}
int feasible(int row,int col)
{
int i,tcol;
for(i=0;i<n;i++)
{
tcol=getmarkedcol(i);
if(col==tcol||abs(row-i)==abs(col-tcol))
return 0;
}
return 1;
}
void nqueen(int row)
{
int i,j;
if(row<n)
{
for(i=0;i<n;i++)
{
if(feasible(row,i))
{
a[row][i]='Q';
nqueen(row+1);
a[row][i]='.';
}
```

```
}
}
else
{
cout<<"\n"<<"The solution is:";
printmatrix();
}
}
int main()
{
int i,j;
clrscr();
cout<<"\n"<<"Enter the no of Queens:";
cin>>n;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
a[i][j]='.';
nqueen(0);
getch();
return(0);
}
```

OUTPUT:

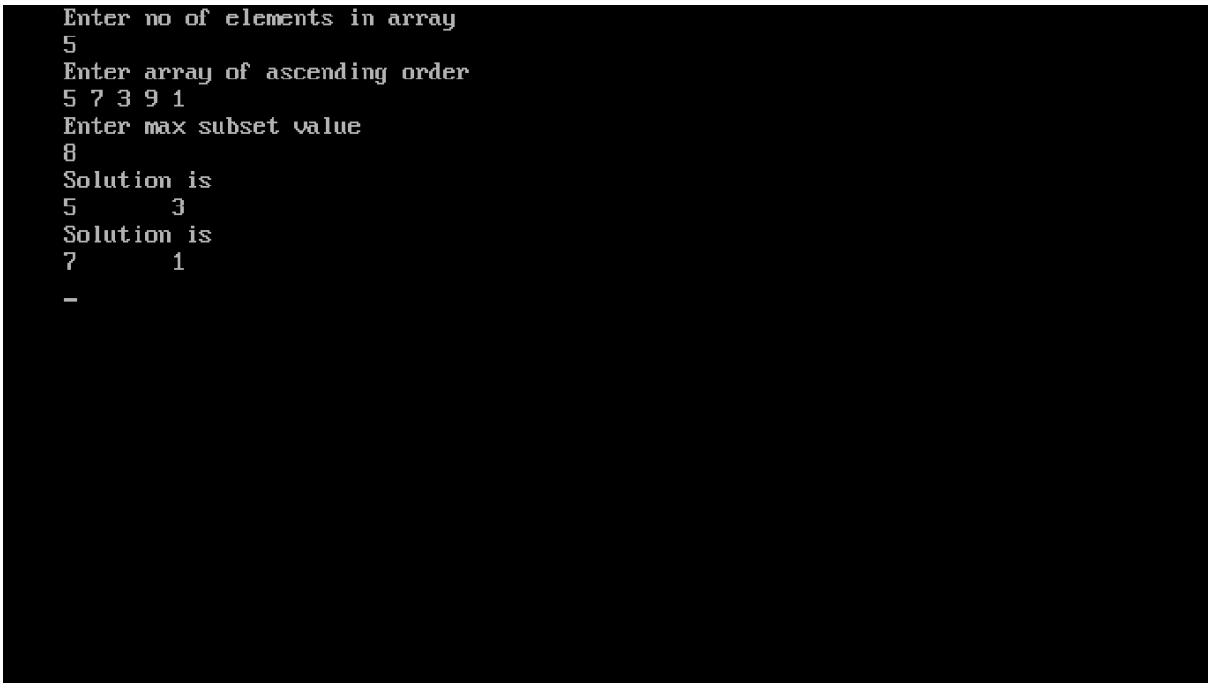

Enter the no of Queens:4
The solution is:
. Q . .
. . . Q
Q . . .
. . Q .
The solution is:
. . Q .
Q . . .
. . . Q
. Q . .

10. Sum Of Subset Of Number

```
#include<iostream.h>
#include<conio.h>
void subset(int n,int w[],int d);
int main()
{
int n,w[10],d,i;
clrscr();
cout<<"Enter no of elements in array\n";
cin>>n;
cout<<"Enter array of ascending order\n";
for(i=1;i<=n;i++)
cin>>w[i];
cout<<"Enter max subset value\n";
cin>>d;
subset(n,w,d);
getch();
return 0;
}
void subset(int n,int w[],int d)
{
int i, s=0,x[10];
for(i=0;i<=n;i++)
x[i]=0;
int k=1;
x[k]=1;
while(1)
{
if(k<=n&&x[k]==1)
{
if(s+w[k]==d)
{
cout<<"Solution is\n";
for(i=1;i<=n;i++)
{
if(x[i]==1)
cout<<w[i]<<"\t";
}
cout<<"\n";
x[k]=0;
}
else if(s+w[k]<d)
{
s+=w[k];
}
else
{
x[k]=0;
}
}
else
```

```
{
k--;
while(k>0&&x[k]==0)
{
k--;
}
if(k==0)
break;
x[k];
x[k]=0;
s=s-w[k];
}
k+=1;
x[k]=1;
}
}
```

OUTPUT:


Enter no of elements in array
5
Enter array of ascending order
5 7 3 9 1
Enter max subset value
8
Solution is
5 3
Solution is
7 1